UNIVERSIDAD RAFAEL LANDÍVAR
FACULTAD DE INGENIERÍA
INGENIERÍA EN INFORMÁTICA Y SISTEMAS
LENGUAJES FORMALES Y AUTÓMATAS
CATEDRÁTICO: ING. MOISÉS ALONSO

FASE 2:
ANALIZADOR SINTÁCTICO

JOSÉ FERNANDO OLIVA MORALES
CARNÉ:1251518

GUATEMALA, 13 DE ABRIL DE 2020

INTRODUCCIÓN

Como parte de la segunda fase del proyecto del curso LENGUAJES FORMALES Y AUTÓMATAS, la cual corresponde al analizador sintáctico de la gramática, siendo continuación del analizador léxico de la misma, partiendo de la expresión regular obtenida de un archivo de texto, de la sección TOKENS, validando que cada uno de los elementos utilizados dentro de dicha sección sean válidos y de ser requerido se encuentren en la sección SETS.

Para ello, se parte del árbol de expresiones armado en la fase previa para posteriormente ser recorrido con el fin de obtener el valor de First, Last y Follow de cada uno de los símbolos. Adicionalmente, se obtiene y presenta la tabla de estados del autómata finito determinista.

Como punto adicional, se muestra una representación gráfica del árbol de la expresión regular obtenida del archivo que contiene la gramática.

OBJETIVOS

General

1. Implementar un analizador sintáctico que permita la lectura y análisis de una gramática a partir de un archivo de texto.

Específicos

1. Obtener una expresión regular a partir de los tokens presentes en el archivo.
2. Obtener el valor de first, last y follow de cada uno de los nodos del árbol de la expresión regular.
3. Obtener la tabla de transiciones para el autómata finito determinista del mismo.

ALGORITMO: OBTENER FIRST, LAST, FOLLOW Y ESTADOS A PARTIR DE UN ARBOL DE EXPRESIONES REGULARES

Por: Fernando Oliva (feroliv4z@gmail.com).

Entradas:

1. generator (lista de símbolos generada a partir de la expresión regular).
2. tree (árbol binario generado a partir de la expresión).
3. simbols (cantidad de símbolos).
4. regex (expresión regular obtenida de tokens).

Salidas:

1. tblFL (lista de símbolos con el first, last y nullable correspondiente).
2. tblFollow (lista de símbolos con el follow correspondiente).
3. tblStatus (lista de estados obtenida a partir de follow).

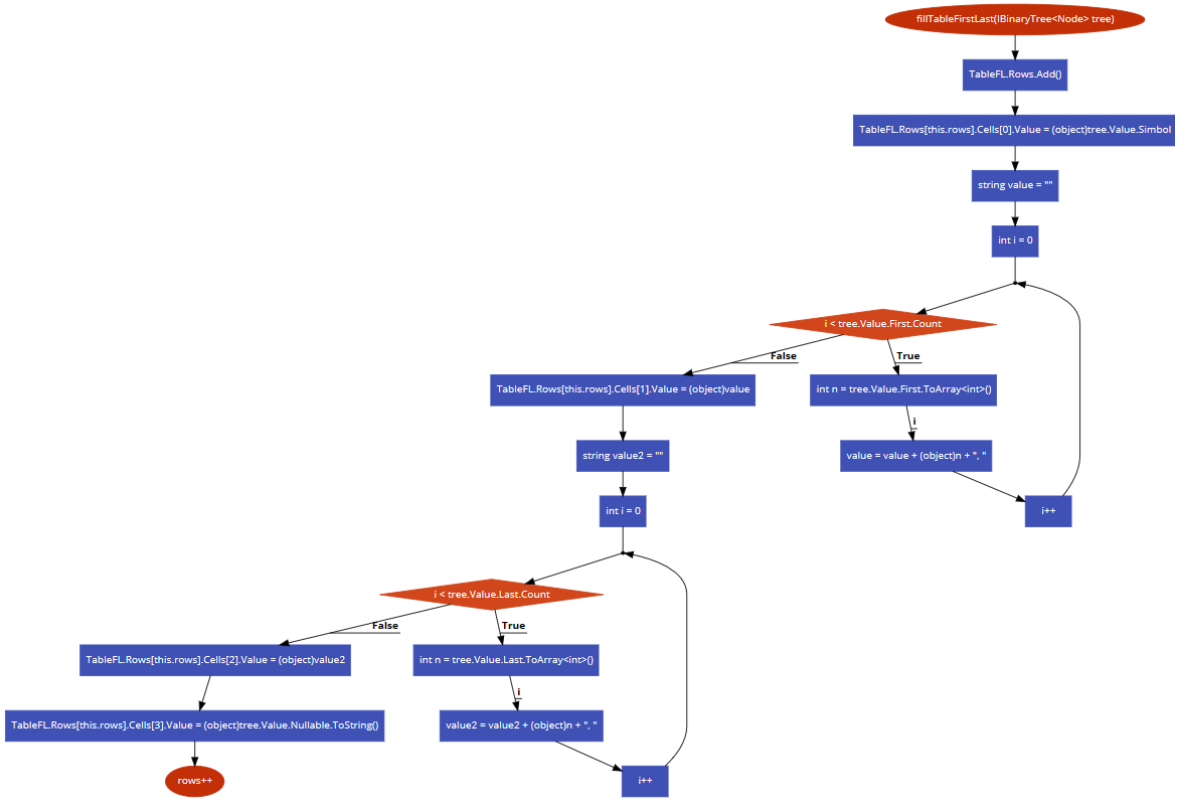
Proceso:

1. Recorrer el arbol de la expresión en post-orden
   a. Para contador = 0, contador < simbols -1, contador++
      i. Si node.value = "*" o "?"
         a. node.nullable = true
      ii. Si no
         a. node.nullable = false
         b Si node.left = null y node.right = null
            i. node.first = contador + 1
            ii. node.last = contador + 1
            iii. tblFL añadir node
         c. Si no
            i. Si node.value = "|"
               a. si node.left.nullable o node.rigth.nullable
                  i. node.nullable = true
               b. node.first = node.left.first + "," + node.right.first
               c. node.last = node.left.last + "," + node.right.last
            ii. si node.value = "."
               a. si node.left.nullable y node.rigth.nullable
                  i. node.nullable = true
               b. si node.left.nullable
                  i. node.first = node.left.first + "," + node.right.first
               c. si no
                  i. node.first = node.left.first
               d. si node.right.nullable
                  i. node.last = node.left.last + "," + node.right.last
               e. si no
                  i. node.last = node.left.last
            iii. si node.value = "+" o "*"
               a. node.first = node.left.first
               b. node.last = node.left.last
2. Recorrer el arbol de la expresión en post-orden
   a. Para contador = 0, contador < simbols -1, contador++

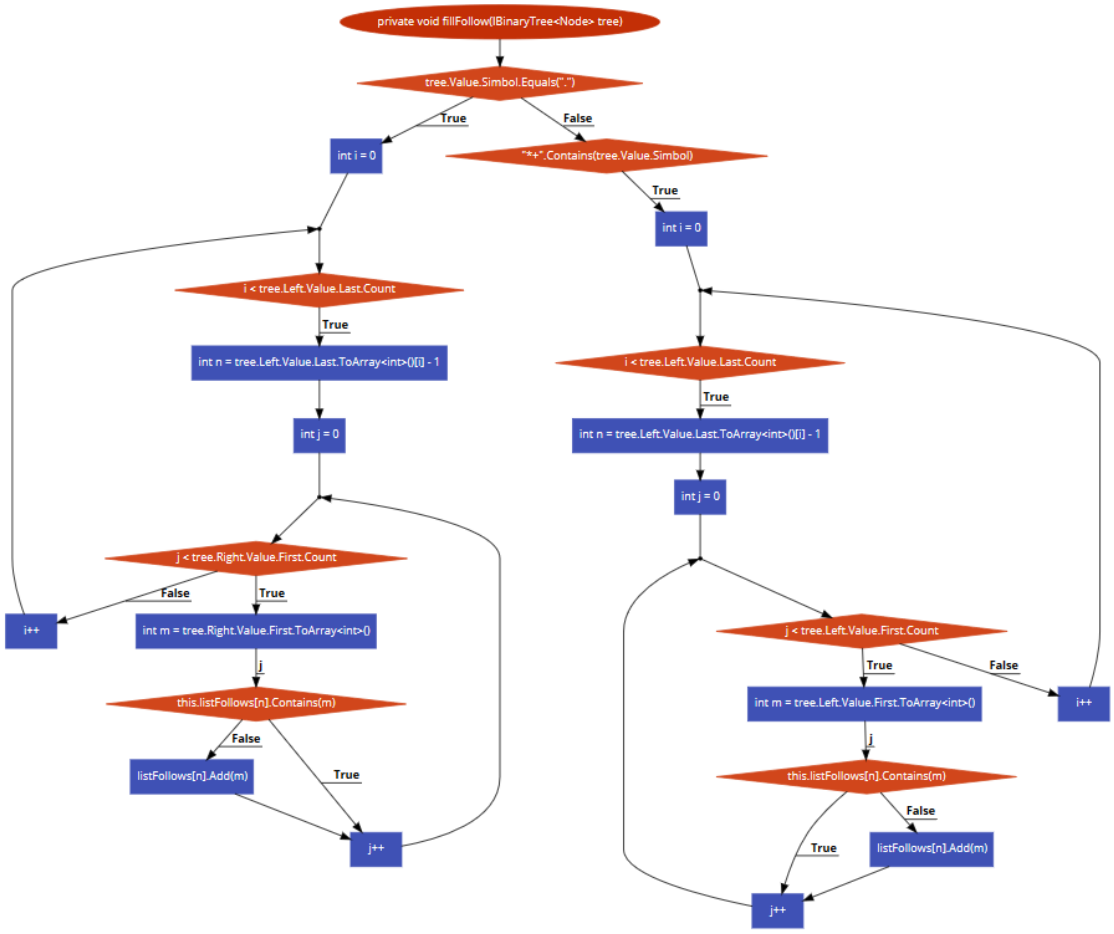
i. si node.left y node.right
a. Si node.value = "+" o "*"
i. Para cada node.left.last
a. node.follow = node.left.first
b. tblFolow añadir node
b. Si node.value = "."
i. Para cada node.left.last
a. node.follow = node.right.first
b. tblFollow añadir node
c. si no
i. node.follow = null
3. Para cada simbolo crear lista en TblStatus para cada estado
4. Si node.parent = null
a. tblStatus añadir node.first
b. mientras estado anterior sea distinto que listasimbolo
i. Para cada node.first
a. si first = simbolo
i. listaSimbolo añadir simbolo
5. Mostrar tblFL, tblFollow y tblStatus

FLUJO DE LOS MÉTODOS

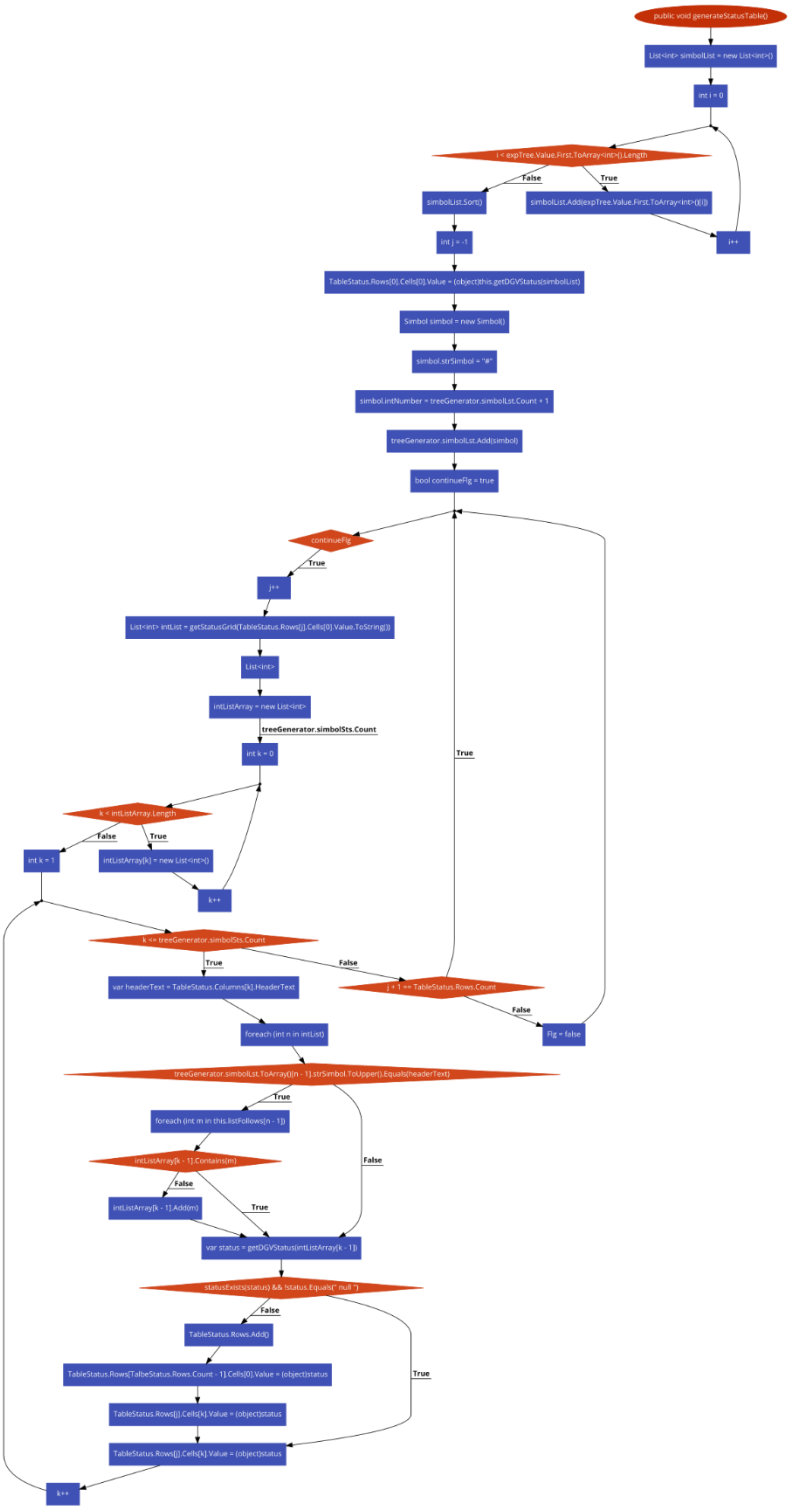
fillTableFirstLast



fillFollow

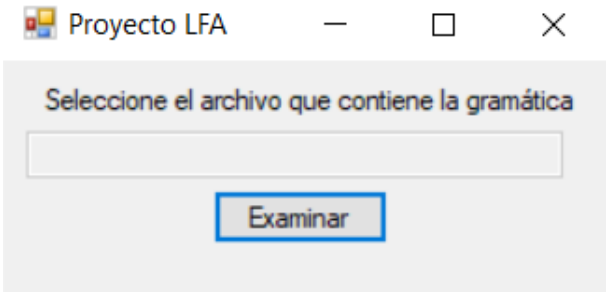

private void fillFollow(IBinaryTree<Node> tree)
tree.Value.Simbol.Equals("ε")
True
False
int i = 0
"*+".Contains(tree.Value.Simbol)
True
int i = 0
i < tree.Left.Value.Last.Count
True
int n = tree.Left.Value.Last.ToArray<int>()[i] - 1
int j = 0
j < tree.Right.Value.First.Count
False
True
i++
int m = tree.Right.Value.First.ToArray<int>()
j
this.listFollows[n].Contains(m)
False
True
listFollows[n].Add(m)
j++
i < tree.Left.Value.Last.Count
True
int n = tree.Left.Value.Last.ToArray<int>()[i] - 1
int j = 0
j < tree.Left.Value.First.Count
True
False
int m = tree.Left.Value.First.ToArray<int>()
i++
j
this.listFollows[n].Contains(m)
False
True
listFollows[n].Add(m)
j++

generateTableStatus

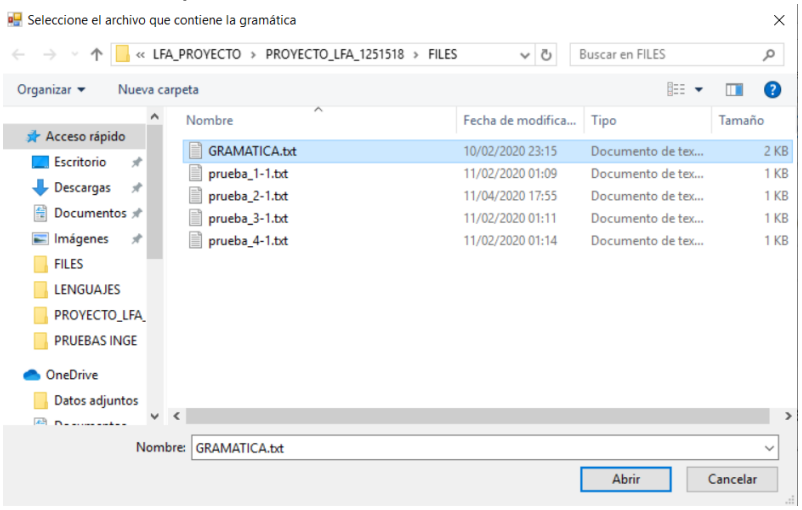


MANUAL DE USUARIO

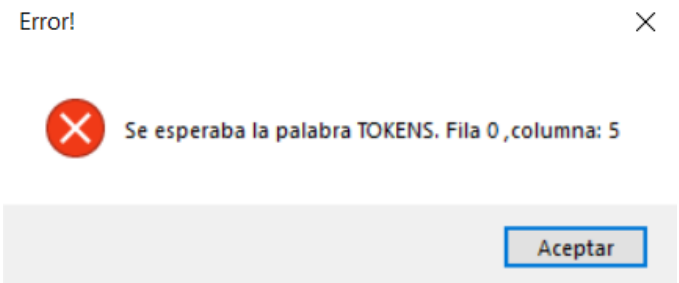
1. Al inicializar el programa, se muestra la siguiente ventana, en la cual se indica que debe seleccionarse el archivo que contenga la gramática que se desee analizar. Debe presionarse el botón “Examinar” para proceder a seleccionar el archivo.



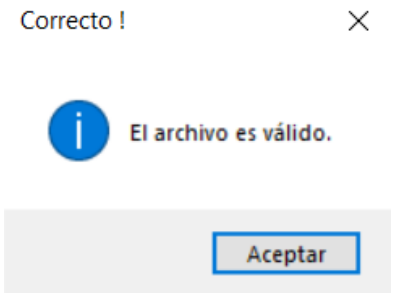
2. Al presionar el botón “Examinar”, se muestra un explorador de archivos en el que se debe seleccionar el archivo deseado. Para ello se requiere presionar el botón “Abrir en el explorador”



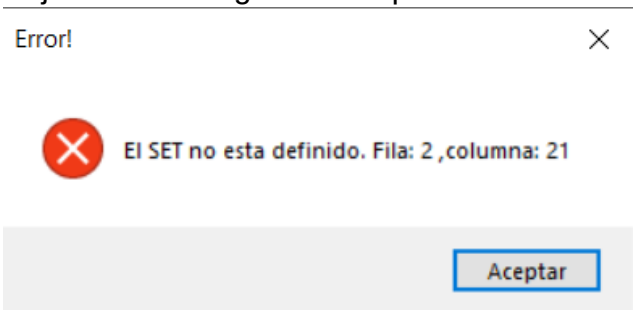
3. Primero, el contenido del archivo seleccionado pasa por el analizador léxico, en el que en el caso de encontrar un error se muestra un mensaje similar al siguiente:



4. Si el analizador léxico da a conocer que el archivo es correcto, se muestra el siguiente mensaje en pantalla.



5. Por su parte, si el analizador sintáctico encuentra un error en el archivo, se muestra un mensaje similar al siguiente en pantalla.



6. Si el analizador sintáctico, da a conocer que el archivo cumple con las reglas correspondientes, se muestra el resultado en pantalla. El resultado incluye, la expresión regular obtenida de la sección tokens del archivo, la tabla de first, last y follow, y la tabla de estos correspondiente. Adicionalmente, se muestra una representación gráfica del árbol de la expresión regular obtenida, para ser visualizado debe desplazarse sobre dicha sección.

BIBLIOGRAFÍA

- Harold Abelson, Gerald Jay Sussman, and Julie Sussman. Structure and Interpretation
- of Computer Programs. MIT Press, 1996.
- Niklaus Wirth. The design of a Pascal compiler. Software - Practice and Experience, 1971.
- Torben Ægidius Mogensen, Basics Of Compiler Design, University of Copenhagen, Denmark, 2000.